VIEW: STATUS REPORT OF A COMPUTER PROGRAM FOR DISPLAY OF ARBITRARY DATA BASES

E. F. Harslem, et al

RAND Corporation

Prepared for:

Advanced Research Projects Agency

August 1972

COMPUTER PRODUCTS GROUP - Pricing Form

CPG - _____ GRA Accession # AD-754089 Date Received _____

One Time Costs			
Normal NTIS Processing $30.00	CPG Evaluation	Subtotal	Predicted No. of Sales (2)
Purchase Price (1)	CPG Announcement	Less Contrib. Subsidy	Unit Cost
Archival	CPG Marketing	Total Cost	Evaluator's Initials

Value of Information		Cost to Fill Each Order	
Tech Report @ 05¢/pg Documentation @ 25¢/pg Data @ 1¢/Kbyte Source Deck @ 1¢/Stmnt Object Deck @ 10¢/card	Qty	Order Processing	Time & Material (4)
	@	CPG Processing	Total PEC CPG
	Value	Shipping & Mailing	Cost Per Input
Multiplier (3)	Adjusted Value	Royalty	Sales Price
Approved by		Date	Fiche PRICE

(1) Explanation of Purchase

See new form

(2) Basis of Prediction

(3) Basis if not Equal to 1

(4) Computation:
Documentation @ 5¢/pg
Cards @ 1¢/card
Minireels @ $10/reel
Regular Tape @ $25/reel

Qty	
@	

NTIS-CPG-73-36 (TEST FORM)

ARPA ORDER NO.: 189-1

R-1069-ARPA
August 1972

VIEW: Status Report of a Computer Program for Display of Arbitrary Data Bases

E. F. Harslem and J. F. Heafner

A Report prepared for

ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC15 67 C 0141. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of ARPA.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY	2a. REPORT SECURITY CLASSIFICATION
The Rand Corporation	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

VIEW: STATUS REPORT OF A COMPUTER PROGRAM FOR DISPLAY OF ARBITRARY DATA BASES

4. AUTHOR(S) (Last name, first name, initial)

Harslem, E. F. and J. F. Heafner

5. REPORT DATE	6a. TOTAL NO. OF PAGES	6b. NO. OF REFS.
August 1972	44	12

7. CONTRACT OR GRANT NO.	8. ORIGINATOR'S REPORT NO.
DAHC15 67 C 0141	R-1069-ARPA

9a. AVAILABILITY/LIMITATION NOTICES	9b. SPONSORING AGENCY
DDC-A	Defense Advanced Research Projects Agency

10. ABSTRACT

A conceptual design and implementation plan for VIEW (Video Information Exchange Window), an interactive computer program designed to interface between a graphics terminal user and his executing program to display and manipulate whatever local or remote data base he may access. VIEW assists in a wide variety of actions. It can plot contour maps and construct original graphics. There are no restrictions on program type or syntax. Default values are provided for all options. Operating in parallel with the user's program, VIEW converses with it through a simply structured data file, the picturefile. The user can freely modify and annotate any display from the terminal, and can obtain hardcopy of any display and a listing of the variables. Designed to cope with the mass of data produced by climate dynamics simulations, VIEW is modular to permit piecewise implementation with user feedback. The device-dependent portion is insulated from the rest, for transferability. PDP-10 implementation is planned.

11. KEY WORDS

VIEW
Computer Graphics
Climate

ARPA ORDER NO.: 189-1

R-1069-ARPA
August 1972

VIEW: Status Report of a Computer Program for Display of Arbitrary Data Bases

E. F. Harslem and J. F. Heafner

A Report prepared for

ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

Bibliographies of Selected Rand Publications

Rand maintains a number of special subject bibliographies containing abstracts of Rand publications in fields of wide current interest. The following bibliographies are available upon request:

Aerodynamics • Arms Control • China • Civil Defense
Communication Satellites • Communication Systems
Computer Simulation • Computing Technology
Decisionmaking • Game Theory • Maintenance • Middle East
Policy Sciences • Probability • Program Budgeting
SIMSCRIPT and Its Applications • Southeast Asia
Space Technology and Planning • Statistics • Systems Analysis
USSR/East Europe • Weapon Systems Acquisition
Weather Forecasting and Control

To obtain copies of these bibliographies, and to receive information on how to obtain copies of individual publications, write to: Publications Department, Rand, 1700 Main Street, Santa Monica, California 90406.

Published by The Rand Corporation

PREFACE

This report presents a conceptual design for a computer graphics program capable of displaying arbitrary data bases. The program was motivated by a need to provide new dimensions in information presentation to researchers studying the dynamics of climate. Sponsored by the Defense Advanced Research Projects Agency, Office of Information Processing Techniques, the new program is an integral part of an overall effort by DARPA and Rand to expand the use of computer resources in military environments.

There is a specific need for the application of advanced graphical display techniques to climatological studies. The main purpose of the report, however, is to serve as a guideline in implementing the program on a PDP-10 computer. Thus the report is primarily directed to the implementer, who is assumed to be familiar with the PDP-10 TENEX operating system and with the BLISS language.

SUMMARY

This is a report of progress in an ongoing ARPA-sponsored project to support research into the dynamics of climate. It describes the planned interactive graphic computer program, VIEW (Video Information Exchange Window), which is intended to interface between a graphics terminal user and his specific application program to display and manipulate whatever local or remote data base he chooses to access. VIEW assists in a wide variety of actions, from laying out a logic display to plotting contour maps and constructing original graphics.

There are no restrictions on the nature and purpose of the application programs that VIEW serves, nor on the syntax the user employs to communicate with the programs through VIEW. VIEW converses with the application program only through a simply structured data file; the program and VIEW operate in parallel. VIEW ordinarily understands the syntax of a user's request but does not interpret the semantics, except for a few direct requests. VIEW's syntax checking can also be bypassed when the user wishes only to use VIEW to pass information to the application program.

To illustrate the way a user interacts with VIEW, several examples are given of using VIEW to create graphs, diagrams, and contour maps and to tabulate numeric data. Following this introduction to the language by scenario, the syntax and semantics of the VIEW "picture language" are explained. The picture language operates on a VIEW-created intermediate data base, called a *picturefile*, not directly on the user's data base. A picturefile contains the data and display codes necessary for one particular picture or other screen display. Picturefiles can be superimposed for display purposes.

To simplify the user's task, a collection of sample system picturefiles is available, with generally acceptable values for graphs, contours, and other displays. A user may retrieve a copy of any desired sample, modify it as desired, and store the result under a new name. In all cases, VIEW provides default values for all display options. Hardcopy of any display can be obtained, as can a listing of the variables.

From the terminal, the user may annotate any display, via the keyboard, light pen, or Rand Tablet stylus. He may also create his own displays by choosing from a menu of displayed options (e.g., line segments, arcs, strings) and touching the points at which each is to appear on the screen, such as the end points of a line. Three levels of intensity are available for any element--dim, normal, or bright. Lines may be solid, dashed, composed of symbols, etc. The user can duplicate, delete, modify, or move any element in a display. A user-defined prototype can be filed and recalled as desired.

To serve as a guideline for the implementer, the program organization and data structure (the picturefile format) are spelled out in some detail. The program is partitioned into functional subroutines to permit piecewise implementation, with later additions in response to user feedback. The special support software for the specific classes of displays is modular and insulated from the main body of VIEW, to allow new modules to be added conveniently. The device-dependent portion of the software package is insulated from the remainder of the program, to make it readily adaptable to different graphic hardware.

The implementation plan for VIEW specifies a PDP-10 computer, using its TENEX operating system and the BLISS language, with Rand Video Graphics terminals. Familiarity with TENEX and BLISS is necessary for complete understanding of the details.

The graph and contour plotting portions, with appropriate network access by external processes, are expected to be complete within 6 months. VIEW is expected to evolve over some time, in response to user needs and reactions.

ACKNOWLEDGMENTS

The authors would like to thank Robert Hoffman, Suzanne Landa, and Michael Warshaw for their help in formulating the graphics program described in this report. The authors would also like to thank Gabe Groner and Robert Anderson for their careful review and comments on this report.

CONTENTS

I. INTRODUCTION

SCOPE OF THE GRAPHICS PROGRAM

The VIEW program (Video Information Exchange Window) is intended to assist graphics terminal users in a variety of actions, from laying out logic displays to plotting contour maps. Basically, VIEW interfaces between the user and his specific application program to display whatever local or remote data base he chooses to access. Remotely located data bases will be accessed via the ARPANET (ARPA Experimental Computing Network).[1]

The design of VIEW was shaped by the criteria of a general-purpose program. As a result:

- There are no restrictions on the nature and purpose of the application processes that VIEW serves, nor on the syntax the user employs to communicate with these processes through VIEW.
- VIEW converses with an application program only through a simply structured data file. VIEW and the application program execute in parallel and communicate only through the data base.
- VIEW ordinarily understands the syntax of a user's requests but, except for a few direct requests, does not interpret the semantics. The meaning of his actions is understood and acted upon by his application program.
- An escape request and a return request enable VIEW's syntax checking to be bypassed if the user wishes to pass information transparently to the application program through VIEW.
- The special support software for creating specific classes of displays, such as x-y plots, is modular and insulated from the main body of VIEW. The resulting program organization allows new modules to be added conveniently.

- The device-dependent portion of the graphics support package is insulated from the remainder of the program, to make it readily adaptable to different graphic hardware.

The implementation plan for VIEW specifies a PDP-10 computer,[2] its TENEX operating system,[3] and the BLISS language.[4] Familiarity with these is necessary for complete understanding of this report.

A CLIMATE DYNAMICS APPLICATION

Although the purpose and scope of VIEW are general, the concept arose from the particular needs of a Rand project that studies the dynamics of climate. This project is heavily dependent upon large simulation and analysis programs. Characteristic of these simulation programs is the production of large volumes of data. One 60-day simulation now produces roughly 8 million numbers. Special analysis programs reduce and manipulate the output of the simulations. The sheer difficulty of handling so much data and the lack of a convenient mechanism to get the data into a form germane to the researcher have hampered progress in this important field of study.

A cursory look at the amount of data associated with just one such simulation program, the Mintz-Arakawa Two-Level Atmospheric General Circulation Model,[5] will illuminate the problem. Data from the simulation runs are maintained on disk packs housing approximately 30 million 8-bit bytes. At present, two packs are required to hold the results of a simulated 60-day experiment. In the near future, the simulated time will increase and, on the average, three disk packs will be needed for each simulation. The number of experiments now stands at six, with eight more likely before the end of 1973. The total number of disk packs for 60-, 90-, and 120-day simulations will be around 50 by that time. This figure could easily go as high as 100 pack-equivalents when the ILLIAC IV[6,7] and the Datacomputer (laser store)[8,9] become available on the ARPANET, making year-long simulations possible.

The climate data base is primarily located at UCLA, with some back-up provided by the University of California, Santa Barbara. In the

future, the bulk of the data will be kept on the Datacomputer at NASA Ames Research Center.

There are now three methods of using the output of the M/A model. (1) A Rand data analysis and reduction program, called MADAT (Mintz-Arakawa Data Package) retrieves arrays of numbers and prints them out. Working with data in this form is perhaps analogous to a programmer perusing a core dump. (2) The results of the MADAT printer listing are hand-plotted as various kinds of graphs. (3) Alternatively, a set of three other Rand programs called MAPGEN, Map Display, and Zonal Profiles generates an intermediate plotting language that can ultimately be used by an FR-80 or other plotter to produce climate maps.(10) About 1400 maps are produced in this way each month. However, MAPGEN is very large, slow, and inconvenient to run, requiring essentially all the capacity of UCLA's IBM 360/91.

VIEW GUIDELINES

It appears that the only long-range solution to this congestion of output is to access the data selectively and configure it dynamically, using an on-line graphics terminal. With this goal in mind, the researchers in the climate dynamics project proffered the following guidelines for the needed graphic support.

The user should be able to:

- Create graphs, contours, bar charts, lists, and vector plots that are user-oriented.*
- Look at a picture and then request hardcopy.
- Easily and automatically generate a standard set of graphs from each simulation's data.
- Request special graphs and plots from the raw data output by the simulations and also from the output of reduction and analysis programs.
- Graph in Cartesian coordinates with both linear and logarithmic scales.

*This report describes the graphics support for graphs, contours, and lists. Subsequent reports will discuss charts and vector plots.

- Scale graphs.
- Produce contour plots with automatic millibar selections.
- See a list of essential parameters, and enter parameters directly by name and value rather than by selection from a large, predetermined menu.

COMPUTER AND DISPLAY EQUIPMENT

Because of the wide variation in the size of programs and the different response-time requirements, the data processing needs of the climate workers are not met by a single computer installation. They now use the IBM 360/91 at UCLA and the IBM 360/75 at UCSB via the ARPANET, along with Rand's IBM 360/65 and PDP-10. They also plan to use the ILLIAC IV. These circumstances meld with the objective of making the graphics program general-purpose to serve other ARPA-sponsored projects at Rand, and perhaps the network community. The equipment around which the VIEW program was planned consists of the Rand PDP-10 TENEX system[3] and the Rand Video Graphics System.[11] However, VIEW may be implemented on different graphic consoles.

PROGRAM GOALS AND IMPLEMENTATION STRATEGY

The overall goals of the graphic support program are:

- Primarily, to provide a graphic facility for researchers and programmers to examine climatological data bases.
- Secondarily, to provide a graphic facility for any ARPA-sponsored projects, at Rand and elsewhere.

The implementation strategy is to offer parts of the program as soon as possible, reflect on their use, and then decide which parts to complete next. The first milestone includes the subset of VIEW (along with retrieval and display processes) needed to support graphs.

II. APPEARANCE TO THE USER

SCENARIOS

It is instructive to examine the user's view of the system from two different directions. First, we provide several scenarios showing the use of the system, without a formal description of the syntax and semantics of the commands and operations being performed. The scenarios give some feeling for the nature of the graphics program. Once this feeling is conveyed, further details of the language for the user of the system are explained, with appropriate mention of options and defaults.

Selecting Data for Analysis

To obtain graphic output of data, a system user must go through two distinct processes: (1) he must retrieve the data to be displayed, and (2) he must specify the parameters required to create the display in the desired format.

Since the graphics program is application-independent, the program retrieving the data is regarded by VIEW as a separate entity. Thus, the graphics program does not examine the inputs to that program. To enter these inputs, the user presses an escape key (Ⓢ) on the keyboard, enters the retrieval parameters, and returns to the normal input mode with another escape key, as shown below.

Ⓢ
{ free-form retrieval requests }
Ⓢ

If the user then types RETRIEVE, the application-specific retrieval process is initiated as a parallel task. The retrieval process may be in the same system or it may be remotely accessed over the ARPANET. When completed, the process returns the selected data to the VIEW graphics program as a data file.

Generating an X-Y Graph

After the desired data have been retrieved, the user may request a graph. If, for example, he enters

```
X = ALTITUDE
Y = PRESSURE
GRAPH
```

he requests that the vector of numbers called ALTITUDE (retrieved data) be plotted agains the vector of numbers called PRESSURE (retrieved data), using the system default values for graphs. On his display screen will appear a graph such as the one shown in Fig. 1.

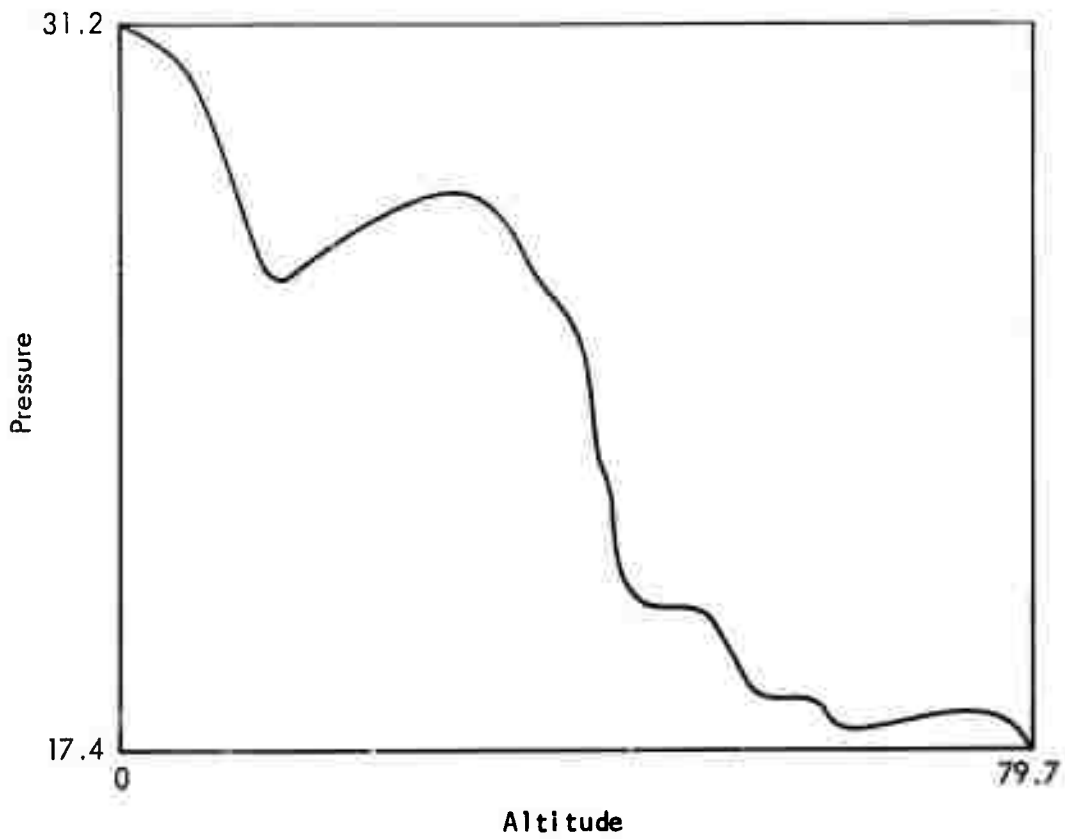


Fig. 1 — VIEW-created graph

By using the default values, the system has fitted the graph exactly to the data; it has used the data extremes to label the x and y axes, and the x and y variable names for the legends on the x and y axes.

If the user wants grid lines, with different limits on the axes, he can enter the statement shown below:

```
XSCALE = 0, 80, 10
YSCALE = 0, 50, 5
GRAPH
```

These requests specify that he wants the x axis to run from 0 to 80 with grid lines every 10 units, and the y axis to run from 0 to 50 with grid lines every 5 units. As a result, he will see a display similar to that shown in Fig. 2.

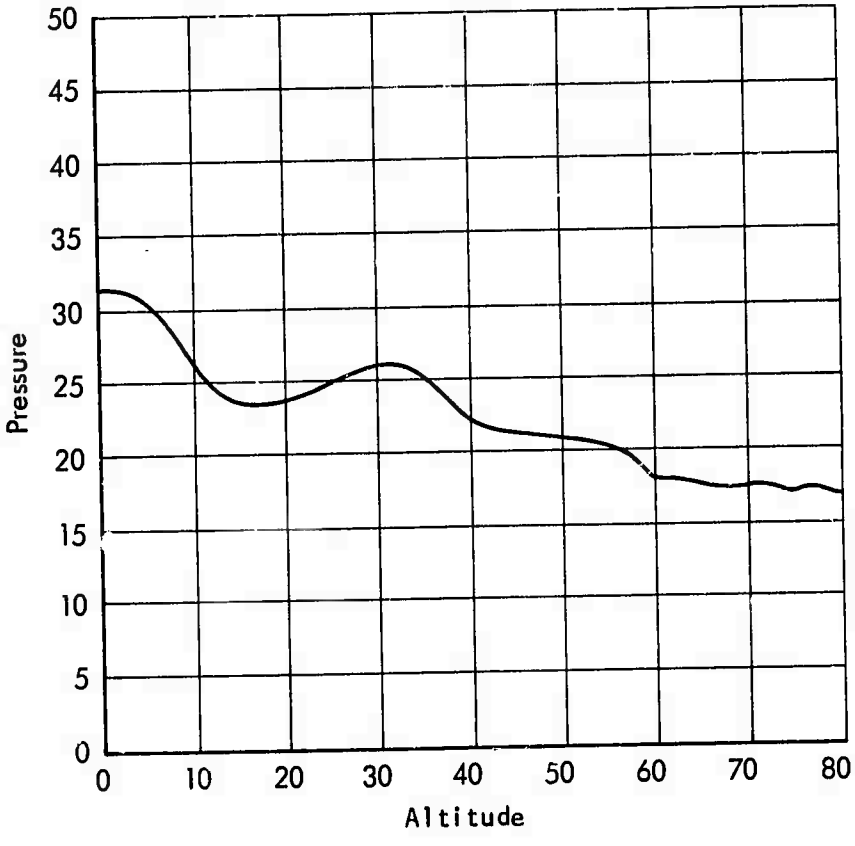


Fig. 2 — VIEW-created graph with grid lines

Generating a Scatter Diagram

Scatter diagrams are similar to graphs, except that there is no necessary dependency relationship between the x and y variables. For example, the user may make the following request:

```
X = PRESSURE, TIME = 400, 435, 5
Y = HUMIDITY
```

In this case, time is the independent variable that is not being plotted. Hence, if the user were to type GRAPH, the result would appear as shown in Fig. 3, which would be of little value as a graph.

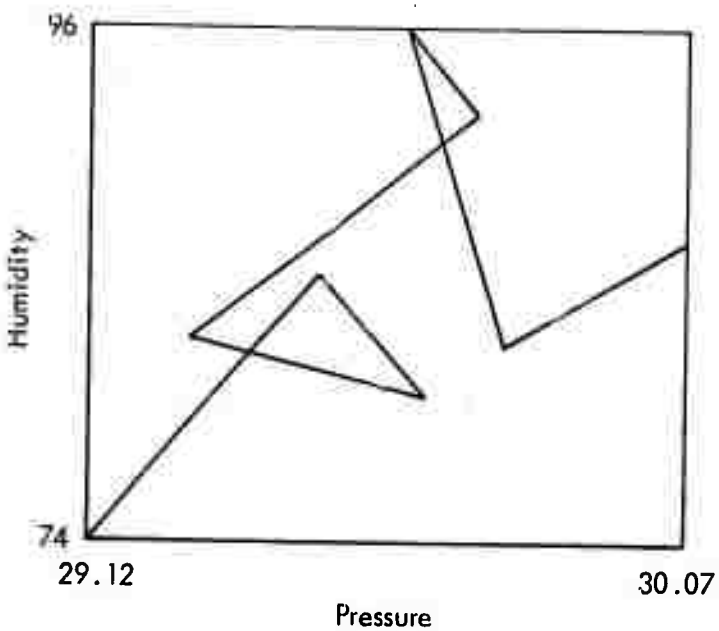


Fig. 3 — Result of plotting independent variables

Since the user really wants a scatter diagram, he will type

```
Y TYPE = +
GRAPH
```

to indicate that the type of plot is the symbol "+" rather than the standard connected line segments. These commands will result in a display similar to that shown in Fig. 4, which is the desired scatter diagram.

Generating a Contour

A typical way for the user to examine three-dimensional information is to view the data in the form of a contour plot. If, for example, his

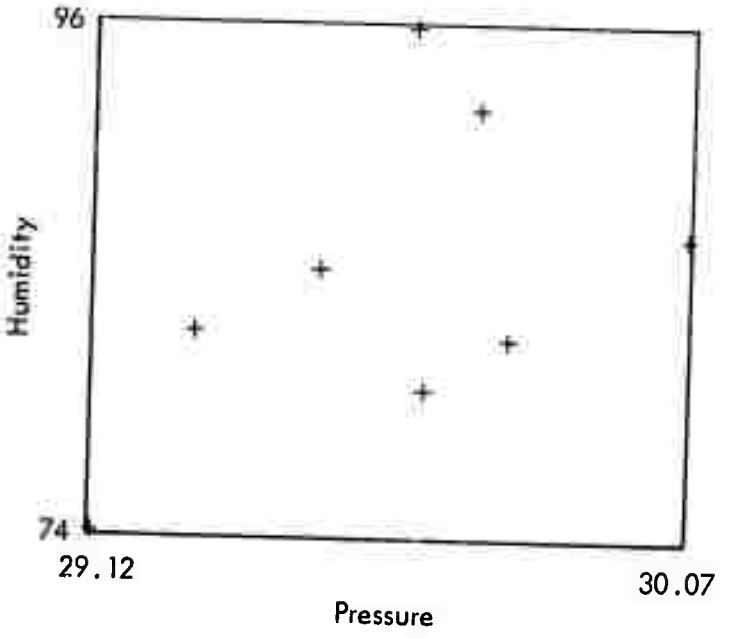


Fig. 4—Scatter diagram of the same points as in Fig. 3

data are an array of pressures over the surface of the earth, with latitude and longitude the two indices, he will probably want to examine the data as a contour plot. Such a display can be obtained by entering the following requests:

```
X = LONGITUDE
Y = LATITUDE
Z = PRESSURE
INTERVAL = 100, 300, 100
CONTOUR
```

These requests specify that he wants pressure as the third dimension and that he wants the contours to run from 100 to 300 at intervals of 100. The result will be the display shown in Fig. 5.

To make the display more informative, the user may want to specify a base contour of, say, 100, and to have that contour dashed instead of solid. He can do this by requesting

```
BASE = 100
BASE TYPE = DASHED
CONTOUR
```

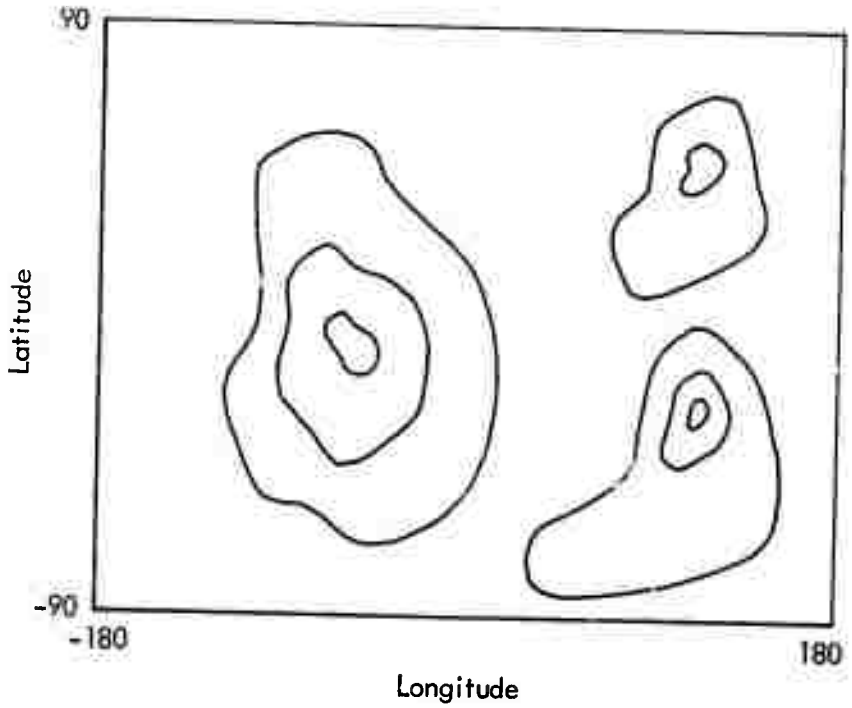


Fig. 5—Contour plot of pressure versus latitude and longitude

This will yield the display shown in Fig. 6, which identifies the 100-level contour more prominently.

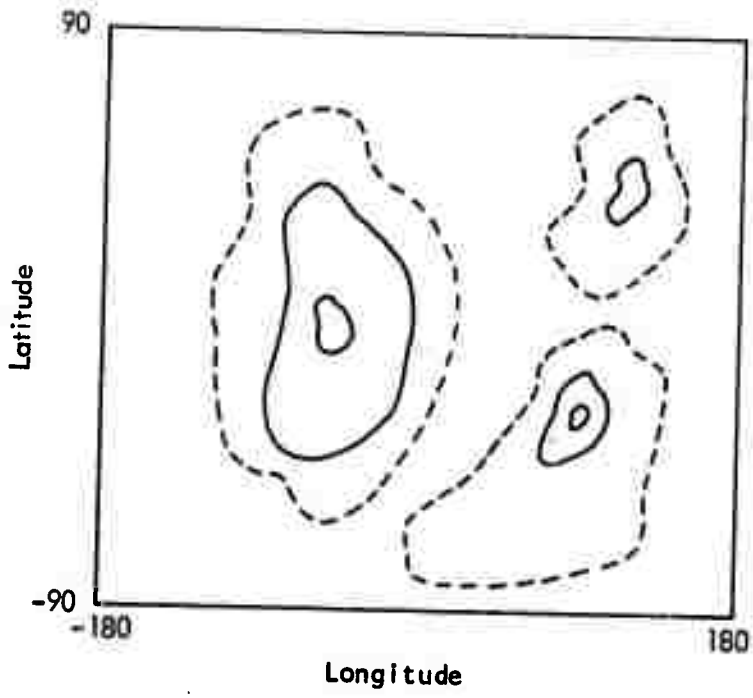


Fig. 6 — Contour plot with dashed base level

Families of Curves

It is often desirable to compare sets of data that have, in the case of x-y graphs, the same x and y variables, but are distinct in a third dimension. Such a display would appear as shown in Fig. 7.

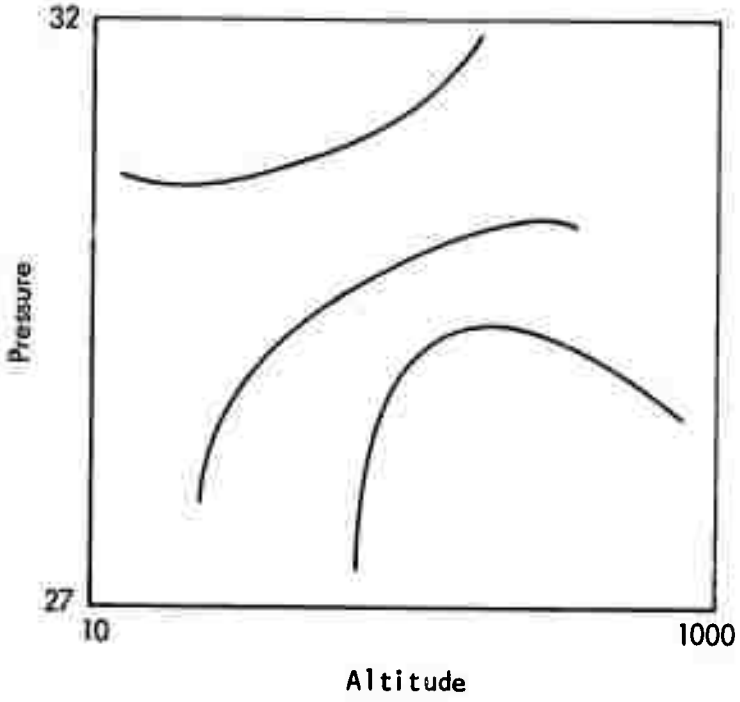


Fig. 7 — Family of curves

This display can be created by (1) superimposing picturefiles or (2) allowing multiple variables specifying the information to be plotted as the dependent variable. In response to user feedback on the initial VIEW system, one of these alternatives will be implemented in the second-generation system.

Generating Lists

Frequently, a user may want to have a scalar, vector, or array displayed numerically, rather than in some graphical format. To specify the manner in which the data are displayed, the user can assign a format descriptor as the value of a variable, e.g.,

variablename = format

The syntax of the format descriptor is similar to that used in JOSS®,(12) with some modifications to satisfy the VIEW requirements.* Within the format descriptor, fields in which numeric data will be stored are indicated by

- A string of underscores with an optional decimal point to indicate standard decimal notation.
- A string of periods to indicate scientific notation.

All other characters may be used to separate fields and to indicate literals. For example, the statement

F = INT = ___, FP = __.__, SN = ...

makes F a variable whose value is a form to display an integer followed by a number with a decimal fraction followed by a number in scientific notation.

The command LIST obtains a display of values in numeric form. The argument of the LIST command is one or more variables. These may be variables with form descriptors as their values, or variables containing the data as specified in the form. If a form is followed by a vector (or vectors) of length N, the form will be applied to the vector(s) element by element. For example, if X and Y are vectors of length 5, then

F1 = X Y

would specify a heading;

F2 = __.- __.-

would specify a two-column format for the numbers; and

*The JOSS format was imitated as a convenience to the users, who are typically familiar with JOSS.

LIST F1, F2, X, Y

would display

X	Y
12.4	77.2
17.5	71.6
19.2	84.2
20.7	99.1
91.4	47.6

The display of a list of numbers is transient; it is generated afresh each time the user requests the display. If a permanent record is desired, the user types the command

HARDCOPY.

LANGUAGE SEMANTICS AND SYNTAX

The Picturefile

To understand the language more easily, it is convenient to look, in general terms, at the data organization. The collection of data necessary to create and display a graph or other picture is called a *picturefile*. A picturefile has a user-chosen name and, as the word picturefile conveys, it contains the display codes for a graph, contour, or other picture. Users can manipulate picturefiles through the picturefile language. A picturefile contains some or all of the following:

- The user-specified variables to retrieve and display a graph or other picture.
- The retrieved data.
- The graphic order codes constructed from the above two kinds of data.
- Any direct inputs from the user at his console (called annotation or comments).

Picturefiles can be superimposed for display purposes. They can be hardcopied; their variables can be listed, etc. Picturefiles are TENEX files; thus, the system's executive file commands apply to them, as well as those commands supported directly by VIEW.

Kinds of Picturefile Variables

A picturefile contains an arbitrary number of *variables*. The kinds of variables are:

- Annotations or user graphics (e.g., joined lines, horizontal lines, vertical lines, character strings, arcs).
- Reserved variables.
- Transparent variables, which VIEW passes to the application program without interpretation.

Annotation variables are input directly by the terminal user, via light pen or stylus, and are understood by an annotation subroutine in VIEW. They provide a way for the user to freely annotate his computer-generated pictures. They can also be used to construct new graphics, as described below, under "Operations on a Picturefile."

Reserved variables, too, are understood by VIEW. They are used to associate one picturefile with another; e.g., they provide forward and backward "links" among picturefiles. They are also used to associate a picturefile with one or more arbitrary processes which, in turn, can understand the transparent variables.

Transparent variables hold no meaning for VIEW. They are acted upon by application-specific processes, such as a retrieval process for Mintz-Arakawa data or a display process for contours. VIEW merely passes them on to the application program, without any attempt at syntax checking or interpretation.

Attributes of Variables

Each variable has a set of attributes that can take on user-assigned values (see Table 1). System defaults are provided for each attribute. Not all attributes will have meaning for a given variable; however, the

attributes exist even with null values. This organization greatly simplifies and generalizes the structural representation of a picturefile.

Table 1

ATTRIBUTES OF A VARIABLE

Attribute	Description
NAME	a string specified by user or process
SPECIFICATION	string
MEANING	string
RESULT	bit string
PICTURE	bit string
TYPE	SOLID, DASHED, TICKS, CHAR (for figures); SMALL, LARGE (for text)
POSITION	x,y coordinates
ORIENTATION	-90 deg ≤ integer ≤ 90 deg
INTENSITY	DIM, NORMAL, BRIGHT
CONTROL	bit string

The user may set or query the value of an attribute. Such operations are performed on the named variable and on the *current picturefile* (a picturefile is made current by retrieving it).

The NAME attribute is the name or label of a variable.

The SPECIFICATION attribute is a character string normally entered by the user. It generally denotes the kind of data contained in the variable; e.g., from the earlier scenarios, where X = ALTITUDE, the character string ALTITUDE is the specification attribute of the variable X.

The MEANING attribute simply defines or describes the variable and is used for tutorial purposes; e.g., the meaning attribute of X might be the character string "DEPENDENT VARIABLE AXIS."

The RESULT attribute contains the numeric data retrieved by an application-specific process.

The PICTURE attribute contains the graphic order codes to produce a display of the numeric data in an appropriate graphical format.

The TYPE attribute indicates the line type or character size for plotting purposes.

POSITION, ORIENTATION, and INTENSITY are display parameters.

The CONTROL attribute is described in Sec. III of this report.

System Picturefiles

To simplify the user's specification task, the VIEW system includes a collection of sample picturefiles with reasonable default values for graphs, contours, etc. A user may retrieve a *copy* of one, modify the copy, and store the result under a new name.

Use of Syntactic Forms

The first syntactic form (variablename) is generally used to set the value of a variable's attribute. Examples are

```
X = TIME, 0, 530, 1
or
TITLE INTENSITY = NORMAL.
```

The second syntactic form (request) is generally used for file requests or to specify a continuing mode of operation. For example:

```
GET GRAPH
or
DISPLAY MENU.
```

The third syntactic form (escapecharacter) allows the user to avoid syntax checking and to enter arbitrary text directly into the picturefile.

The arbitrary text is entered as the value attribute of a globally known variable. The user then returns to the first two syntactic forms by entering a second escape character. The escape feature provides two benefits: The user can communicate with an application process in whatever language is convenient for him, and can couple very powerful external processes to the graphics terminal.

In the variablename and request syntactic forms, the attributename, request, and argumentlist may be abbreviated as the leftmost character string that will identify them uniquely. For example, the request "GET" may be written as "G" if there are no other requests beginning with the letter G.

Operations on a Picturefile

These operations may be most conveniently expressed in tabular form (see Table 2).

Additional VIEW operations on picturefiles are envisioned to support other user-desired functions--most notably, functions for superimposing two or more picturefiles for comparison of sets of data.

Operations on a Variable of a Picturefile

Each picturefile consists of a set of variables that define it. Each variable is in turn defined by the values of its attributes. All variables may be changed through assigning values to their attributes. New variables may be created and old ones purged. All attributes are identified by reserved words in the VIEW language, and some attributes may only take on reserved words as values (see Table 3). Values of attributes initially assume default values, but the values may be examined and changed. Some attributes of some variables may be null.

Table 3 provides examples of these operations.

Reserved Variables

There are two classes of reserved variables--those that are reserved globally, regardless of the function being performed, and those that are reserved in the context of the intended display function (graphs, contours, etc). There are four global variables:

Table 2

OPERATIONS ON A PICTUREFILE

Operation	Result
GET picturefile	Retrieves a copy of the picturefile, which becomes the current picturefile. All subsequent operations on variables affect this image of the picturefile named
PUT picturefile	Saves current picturefile under the name given
PURGE, RENAME, DIRECTORY } picturefiles	Self-explanatory operations, initially supported by the TENEX system
RETRIEVE	Invokes a retrieval or computational process and points it at the current picturefile
GRAPH	Invokes a general-purpose graph-drawing program to operate on the picturefile and produce the display order codes for a graph
CONTOUR	Invokes a general-purpose contour plotting routine to operate on the current picturefile and produce the display order codes for a contour map
CHART	Invokes a histogram program to produce display orders for a bar chart from the current picturefile
LIST	Invokes a list program to produce display order codes for a tabular listing of retrieved data in the current picturefile
PLOT	Invokes a vector plot program to produce display order codes for a vector plot from the current picturefile (not illustrated in this report)

Table 3

OPERATIONS ON ATTRIBUTES OF VARIABLES

Example	Result
X = TIME	Assigns the character string "TIME" to the specification attribute of X. Note that X = TIME is equivalent to X SPECIFICATION = TIME; i.e., if no attributename is given, the specification attribute is assumed. If X is not defined, then a new variable, X, is created.
X =	Sets the specification attribute of X to null. If X has not been defined, then a new variable, X, is created.
X ?	Displays the contents of the specification attribute of X.
X TYPE ?	Displays the currently assigned type attribute of X.
DELETE X	Purges the variable X from the picturefile.
OPTIONS X	Displays the permissible values of the specification attribute of X.
OPTIONS TYPE X	Displays the permissible values of the type attribute of X.

- NEXT and PREVIOUS are used to doubly chain picturefiles so that sets of displays can be logically connected. The specification attribute of these variables should name the appropriate file.
- PROCESS indicates the program that will retrieve data or post-process a picturefile. Its specification attribute is the name of the program.
- MESSAGE contains in its specification attribute any text that an external process wishes to pass back to the user through VIEW via the picturefile.

The second or local class of reserved variables is much larger. Those required for the display of x-y graphs have been specified and are explained below:

- x--The name attribute is the variable to be plotted on the x-axis.
- y--The name attribute is the variable to be plotted on the y-axis. The type attribute of y specifies whether the data points should be connected with straight-line segments or plotted with a symbol at each coordinate pair. The intensity attribute indicates the intensity of the plotted information--dim, bright, or normal.
- XLABEL and YLABEL--The specification attribute specifies the label for the appropriate axis. The intensity, size, position, and orientation attributes have the obvious meanings.
- TITLE--Same as XLABEL, YLABEL, except that it represents the title of the display.
- XSCALE and YSCALE--The specification attributes indicate the limits of the appropriate axis in the user's data space. The specification attribute may also indicate the divisions for each axis.
- GRID--The type attribute indicates whether the axis divisions should be shown with tick marks or grid lines. The intensity indicates the intensity of the entire grid.
- WINDOW--The specification attribute indicates the part of the screen to be used for the display in terms of a grid of 1024 × 1024 "raster units."

Experience will probably dictate when to add to this set of variables to satisfy new requirements. For example, several curves might be required on a single grid, in which case some method of "subscripting" the variables x and y would be appropriate.

Operations on a Picture

A displayed picture may be manipulated through keyboard requests and through user-entered graphics. The user can enter graphics by selecting from a displayed list (menu) of possible options. He selects the operation or graphic element by pointing at its name on the displayed list with a light pen or tablet stylus. (The program will interpret inputs from both of these devices.) He then enters the other points necessary to construct the graphic, such as the end points of lines.

The envisioned menu of alternative constructs includes the following:

- Primitives: strings, "ink," arcs, lines.
- Qualifiers: large characters/small characters; vertical/horizontal/skewed lines; form/solid/dashed lines; lines made up of symbols; dim/normal/bright intensity.
- Compounds: graphic structures defined by the user.
- Primitive and/or compound operators: duplicate; delete; define; name; move.
- Global menu operator: erase screen; scroll compound list; relocate menu; hardcopy.

The menu is an implicit part of each picturefile--i.e., it will be added to, or removed from, the current display at the user's request. When the menu is displayed, the user can select functions to add to the current picture, or can compose an entire picture of nothing but graphics entered by him. The compound function mentioned above allows the user to define a prototype graphic, save it, and later reproduce instances of that prototype.

III. PROGRAM ORGANIZATION AND DATA STRUCTURE

DATA STRUCTURE AND PROGRAM MAPPING

In this section, we use a collection of diagrams to illustrate the planned organization of the VIEW program and the associated data structures. VIEW is implemented in BLISS.[4]

Figure 8 specifies page addresses of program and data, and core addresses of picturefile components. The VIEW program and data proper precede the current picturefile. The picturefile is divided into the five segments shown in Fig. 8 for ease of implementation. The fixed boundary addresses are quite reasonable in relation to the expected amount of data comprising any of the sections. Each of the five segments begins with a two-word segment header of the form shown in Fig. 9. This contains typical header information: the location and the maximum and current lengths.

Figures 10 through 13 show the composition of a picturefile variable. Figure 10 is a detailed illustration of that part of the program variable given in address space 400000-423777 in Fig. 8. Figure 11 shows the variable length specification and meaning attributes, each of which is terminated by a zero byte. Figure 12 shows the structure of the picture attribute of a variable; intensity, position, and type are of fixed length, and the CRT order codes that define the picture segment are of variable length, a function of the picture element being described. Figure 13 shows the result attribute of a retrieval variable.

PROGRAM ORGANIZATION

VIEW is planned to be modular to allow incremental implementation, as well as to facilitate debugging and modification. External programs (users' application programs) interface only through data in picturefiles.

The principal modules of the program are shown in Fig. 14. The *pre-command decoder* accepts and parses the initial part of the user's input (via light pen, stylus, or keyboard) to determine the appropriate

Item	Octal Address
BLISS program variables	
Unused	
BLISS program	
Picturefile variables excluding those below	400000-423777
Specification and meaning attributes of picturefile variables	424000-447777
Video picture attributes of picturefile variables	450000-473777
Video annotation attributes of picturefile variables	474000-517777
Result attributes of picturefile variables	520000-777777

Fig.8 — Virtual core addresses of program and data

Segment start address	Segment's current ending address
Segment's potential maximum ending address	Segment's maximum permissible length

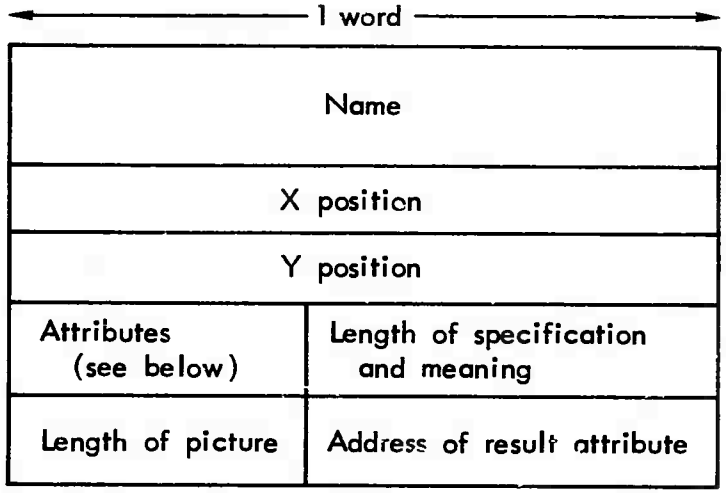
Fig. 9—Header of each picturefile segment

module to invoke. The *picturefile accessor* reads picturefiles from the TENEX file system into the virtual core in the current picturefile location, and also writes the current picturefile into the TENEX file system. The *picturefile editor* is responsible for editing entries in the current picturefile according to the user's requests entered through the keyboard. A typical request is to change the value of one of a variable's attributes.

The *annotator*, similar in function to the picturefile editor, responds to user's requests containing light pen data and possibly text. Typical requests involve decoding menu selections and drawing primitive graphic elements. The *executor* acts on requests for displays or the retrieval of data. It determines (from the current picturefile and the request) which process to invoke, and invokes that process as a separate fork of VIEW. (A fork in TENEX is a parallel process.)

Retrieval programs use the retrieval variables in the current picturefile (or a named picturefile) to selectively retrieve data from a user's data base. These are application-dependent programs provided by the user.

The *display generation programs* (e.g., graph, contour, etc.) use the raw data supplied by a retrieval program (result attributes of retrieval variables), along with values of display variables in the current picturefile, to create CRT displays. As currently envisioned,



Attributes

Bits 0-7 Symbol	Bits 8, 9 Control	Bits 10-13 Type	Bits 14, 15 Intensity	Bits 16, 17 Orientation
Character code	8 = 0, normal picture 8 = 1, annotated picture 9 = unused	0 = undefined 1 = solid 2 = dashed 3 = tick mark 4 = large char. 5 = small char. 6 = symbol	0 = undefined 1 = normal 2 = dim 3 = bright	0 = undefined 1 = horizontal 2 = vertical

Fig. 10—Basic picturefile variable

Variable length character string representing specification attribute	zero byte
Variable length character string representing meaning attribute	zero byte

Fig. 11—Specification and meaning attributes

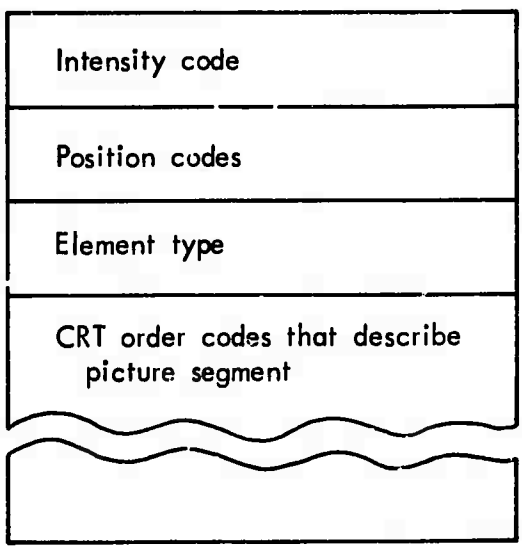


Fig. 12—Picture attributes

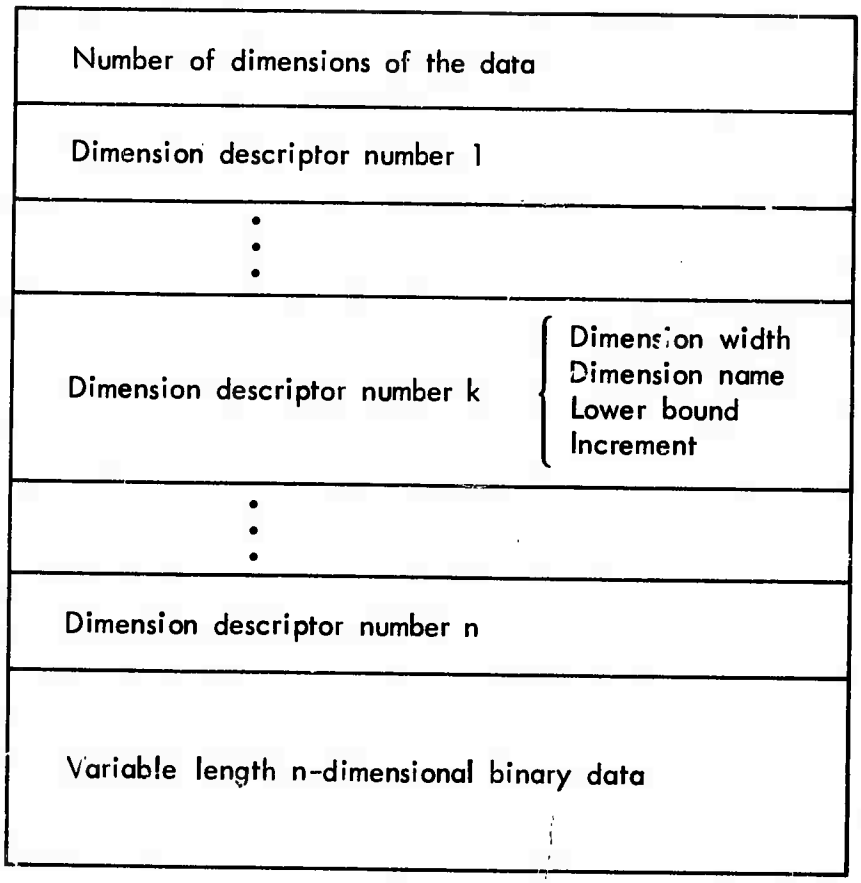


Fig. 13 — Result attribute. The dimension descriptor can be addressed by name by the user through a selector function, which allows him to choose either a row or a column vector from an array for graphing. For example, Y = P selects a column vector P to graph against an index specified by the descriptor element that heads the result attribute of P.

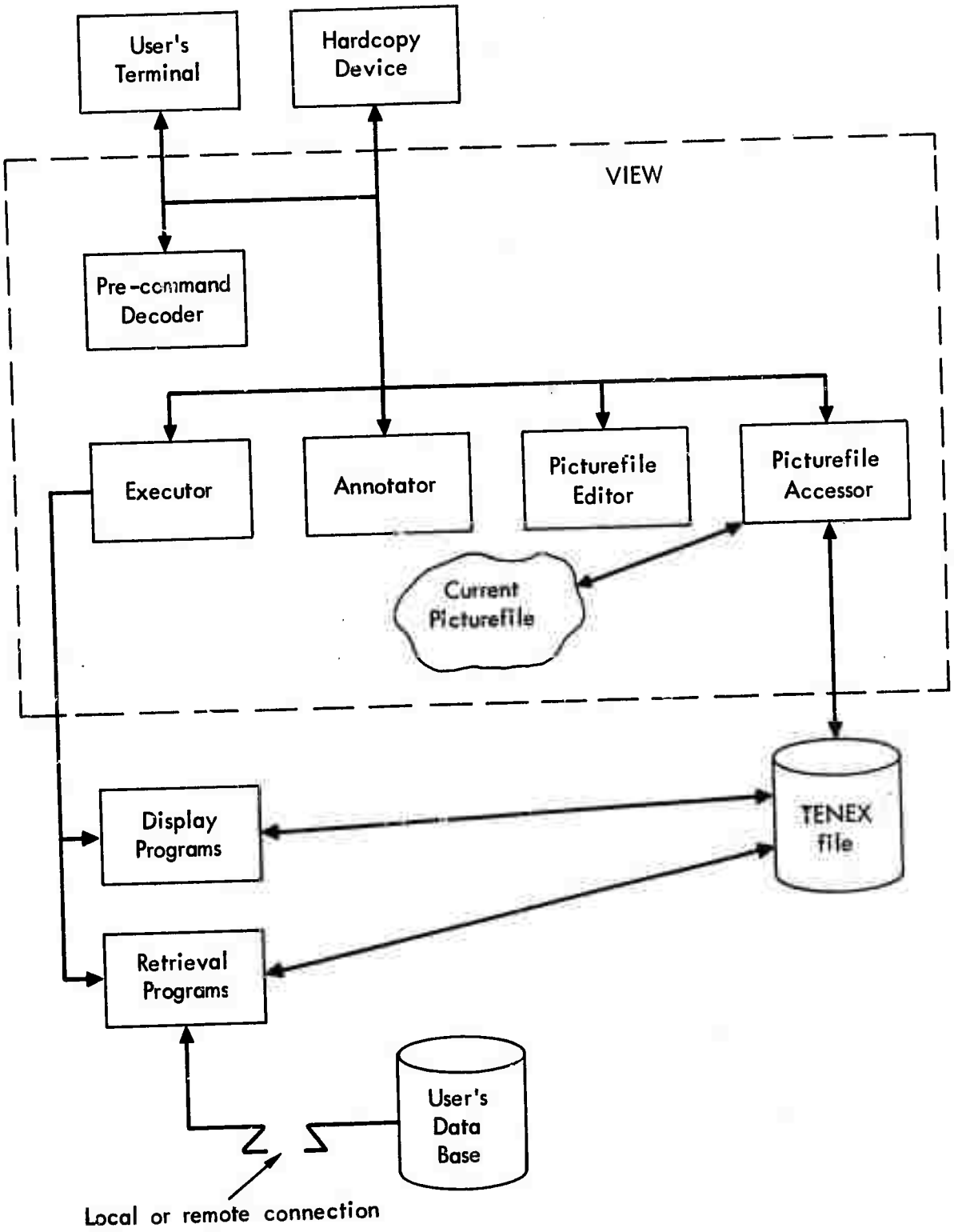


Fig. 14—Basic program organization

each one of these programs will be capable of generating a different class of displays.

Figure 15 shows the interaction of these programs through the current picturefile in simplified form.

INTERFACE WITH EXTERNAL PROGRAMS

The executor module shown in Fig. 14 invokes external processes. In the climate dynamics applications, these processes retrieve raw or reduced data for graphing, contouring, etc. In general, their function might well be to analyze data entered through the keyboard, rather than to retrieve data; therefore, we shall refer to external programs simply as "processes" rather than as "retrieval processes."

The (external) process will be a started as a TENEX fork, inferior to the VIEW program. The name of the TENEX save file containing the process must be the specification attribute of the reserved variable, PROCESS.

Suppose, for example, the user is working with a copy of a picturefile called GRAPH1.MINE;4 (see Fig. 16) and that he types RETRIEVE. The executor module writes the current picturefile (CPF) as filename.extension;versionnumber, using the next highest version number--GRAPH1.MINE;5 in the example. The executor then (1) extracts the value attribute of the PROCESS variable as the save-file name of the process to be initiated; (2) creates an inferior fork to the process; (3) maps the program into the inferior fork's virtual core; (4) loads the registers with the picturefile name GRAPH1.MINE;5 and then (5) starts the fork.

The inferior fork reads GRAPH1.MINE;5 and performs its analysis or retrieval; it then writes its output as GRAPH1.MINE;6, and terminates. Typically, the contents of GRAPH1.MINE;6 will be result attributes and perhaps the specification attribute of the variable MESSAGE. The process can, of course, write anything that it and the user understand.

The executor is interrupt-initiated upon termination of the inferior fork. It reads the value of each attribute of each variable in GRAPH1.MINE;6 and writes those values into the CPF. The executor receives a success or failure return code in the registers, which it relays to the user along with any message from the specification attribute of the variable MESSAGE. The executor then deletes GRAPH1.MINE;5 and GRAPH1.MINE;6.

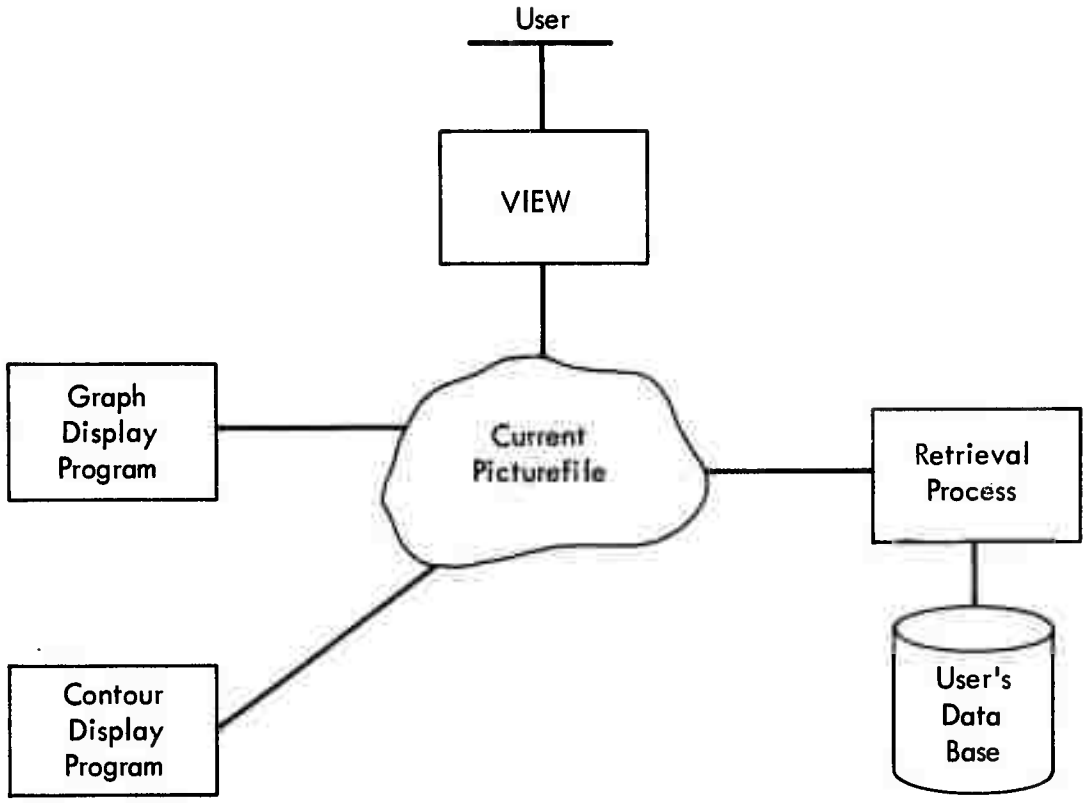


Fig. 15—Program interface via current picturefile

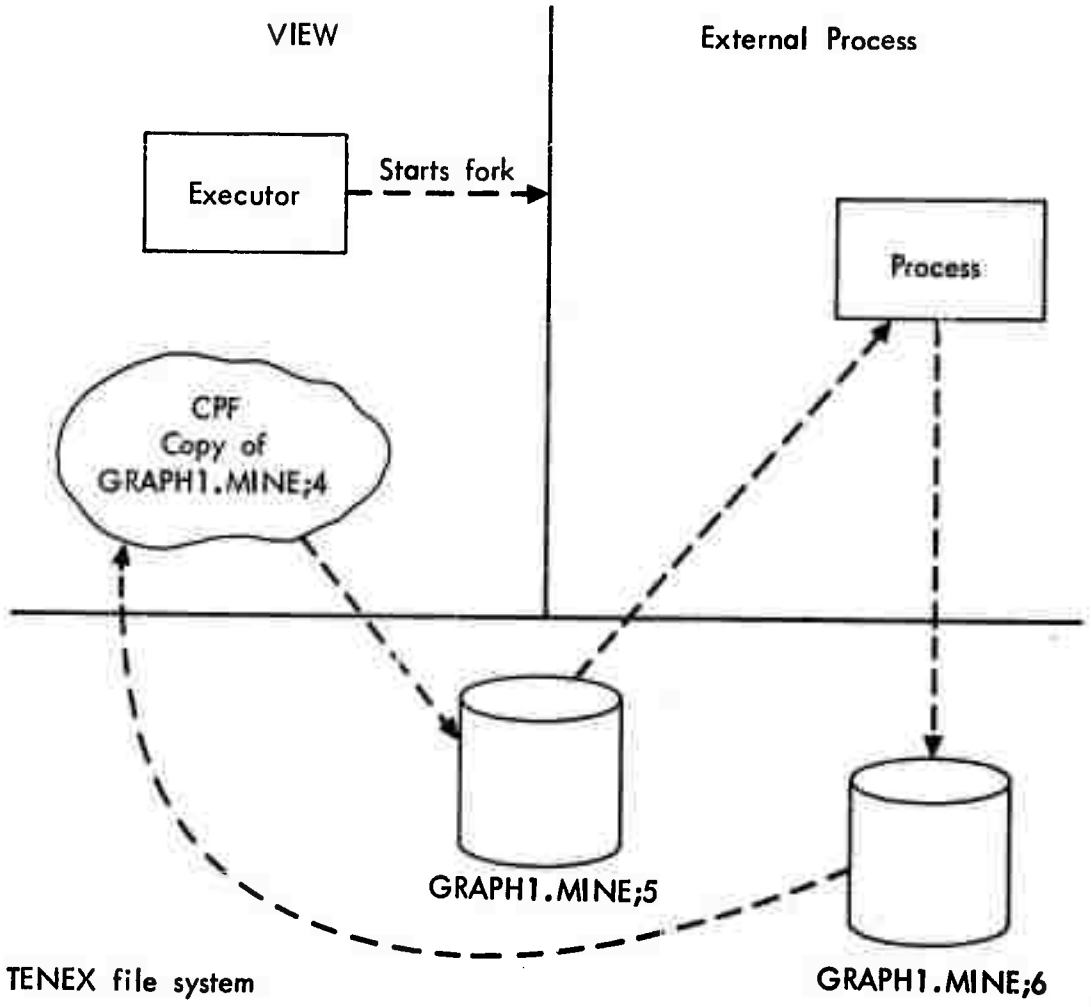


Fig. 16—Working with current picturefile

One restriction applies where the external process is working on the CPF. Normally, where the CPF is used, the user wishes to start the retrieval and analysis while he sets display variables. The restriction is that he cannot write (PUT) the CPF until the inferior fork has terminated. This resolves ambiguities such as the following, which might otherwise occur:

```
GET    Q.M;1
RETRIEVE
PUT    R.M;1
(inferior fork terminates)
```

Is Q.M;3, which the process created, merged into Q.M;1, CPF, or R.M;1?

A SUBPROGRAM PACKAGE FOR GRAPHIC CONSTRUCTION

VIEW was intended to run on the PDP-10 with Rand Video Graphics System[11] consoles, but may be implemented on other graphical consoles. Construction of graphic order sequences for displays occurs in many places within VIEW besides the various display routines (GRAPH, CONTOUR, etc.). This construction is centralized in a subprogram package containing the necessary functions for the entire graphics system.

To enable the user to request displays from the system, create displays, and edit existing displays, the following requirements were included in the subroutine package:

- The ability to create the display orders for strings of text in both horizontal and vertical orientations.
- The ability to create three classes of lines: (1) discrete line segments specified by end points; (2) joined line segments specified by end points; (3) joined line segments generated using incremental vectors for curve following and contouring.
- An editing feature to operate on existing graphic elements. This includes modifying (1) the intensity of an element; (2) the starting position of an element; (3)

the line type or character size of an element; (4) the contents of an element (characters or line segments).

- A function to collect and transmit all or part of existing display orders to the terminal's graphic display hardware for presentation to the user.

This package was planned to be written in BLISS with entry points global to the entire graphics system.

IV. DISCUSSION

Although the graphics data presentation system plan we have described here is an initial one, the system design, in the main, is fairly general. However, some specific references are made to implementation on a PDP-10 TENEX system using Rand Video Graphics consoles. The selection of different consoles for the system does not greatly alter the plans in this report.

Initially the VIEW system will be implemented to use the MADAT program for data retrieval. However, the future availability of the Datacomputer and ILLIAC IV via the ARPANET indicates that analysis and presentation of remotely located data will become more important.

In general, VIEW is expected to evolve over time. The design of the functions for x-y graphs has been reviewed and accepted by the potential users and will be implemented first. It is expected that the graph and contour portions of VIEW will be implemented, with appropriate network access by external processes, within the next 6 months.

In response to user feedback, the plans described here will be modified and the scope of the program expanded, as necessary. We expect eventually to encompass a large set of data representation techniques covering several fields of application.

REFERENCES

1. Roberts, L. G., "Computer Network Development to Achieve Resource Sharing," *AFIPS Conference Proceedings*, Vol. 36, 1970, pp. 543-550.

2. *PDP-10 Reference Handbook*, Program Library, DEC, Maynard, Massachusetts 01754.

3. Bobrow, Daniel G., "TENEX, a Paged Time Sharing System for the PDP-10," *Communications of the ACM*, Vol. 15, No. 3, March 1972, pp. 135-143.

4. Wulf, W. A., et al., *BLISS Reference Manual*, Computer Sciences Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, January 15, 1970.

5. Gates, W. L., et al., *A Documentation of the Mintz-Arakawa Two-Level Atmospheric General Circulation Model*, The Rand Corporation, R-877-ARPA, December 1971.

6. Barnes, G. H., et al., "The ILLIAC-IV Computer," *IEEE Transactions on Computers*, Vol. C-17, No. 8, August 1968, pp. 746-757.

7. Kuck, D. J., "ILLIAC-IV Software and Applications Programming," *IEEE Transactions on Computers*, Vol. C-17, No. 8, August 1968, pp. 758-769.

8. McFarland, K., and M. Hashiguchi, "Laser Recording Unit for High Density Permanent Digital Data Storage," *AFIPS Conference Proceedings*, Vol. 33, Part 2, 1968, pp. 1369-1380.

9. *Datalanguage*, Working Paper No. 3, Computer Corporation of America, Cambridge, Massachusetts, October 29, 1971.

10. Pass, David C., *Output Programs for the Mintz-Arakawa Two-Level Atmospheric Model*, The Rand Corporation, R-1046-ARPA (in preparation).

11. Uncapher, K. W., *The Rand Video Graphic System--An Approach to a General User-Computer Graphic Communication System*, The Rand Corporation, R-753-ARPA, April 1971.

12. Shaw, J. C., "JOSS: A Designer's View of an Experimental On-Line Computing System," *AFIPS Conference Proceedings*, Vol. 26, 1964, pp. 455-464. Also published as a Rand Corporation Paper, P-2922, August 1964.